

---

# Telemetry Subsystem

## Jules Junker

---

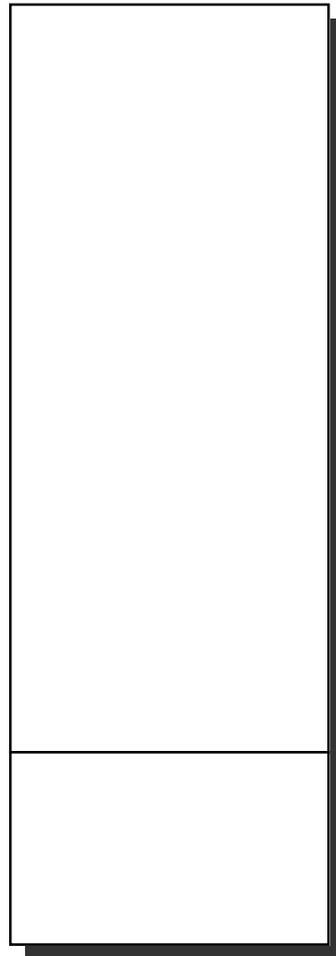
17 October 1995

# Real-time Task Distribution Process Map

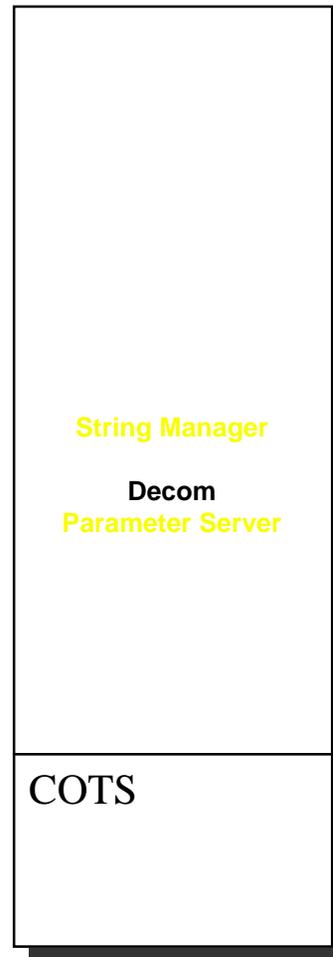
Real-Time Server



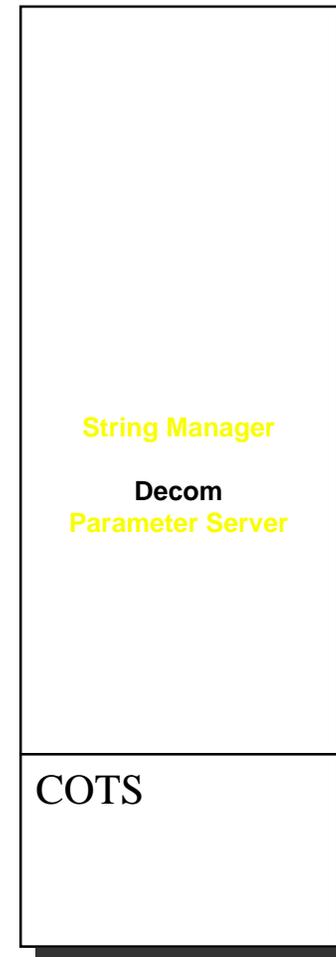
Data Server



User Station



IST



# Telemetry Subsystem Design Features

---

**The Telemetry Decommutation process runs on the RTS, the User Station, and the ISTs**

- **Multicast EDUs can be received on any workstation with no additional load to the RTS**
- **Additional users monitoring data do not impact other systems**
- **Decommutation processes can be adjusted to meet the needs of the user**
  - **Allows users to selectively decommutate any or all parameters without affecting the RTS**
  - **User can adjust limit values and EU coefficients**
  - **User can select limit groups and EU conversion algorithms**

# Telemetry Subsystem Design Features (continued)

---

**Decommutation is stream oriented**

- **Separate decommutation processes are started for each kind of data (Housekeeping, Health & Safety, and Standby)**

**Robust design isolates the failure of one stream**

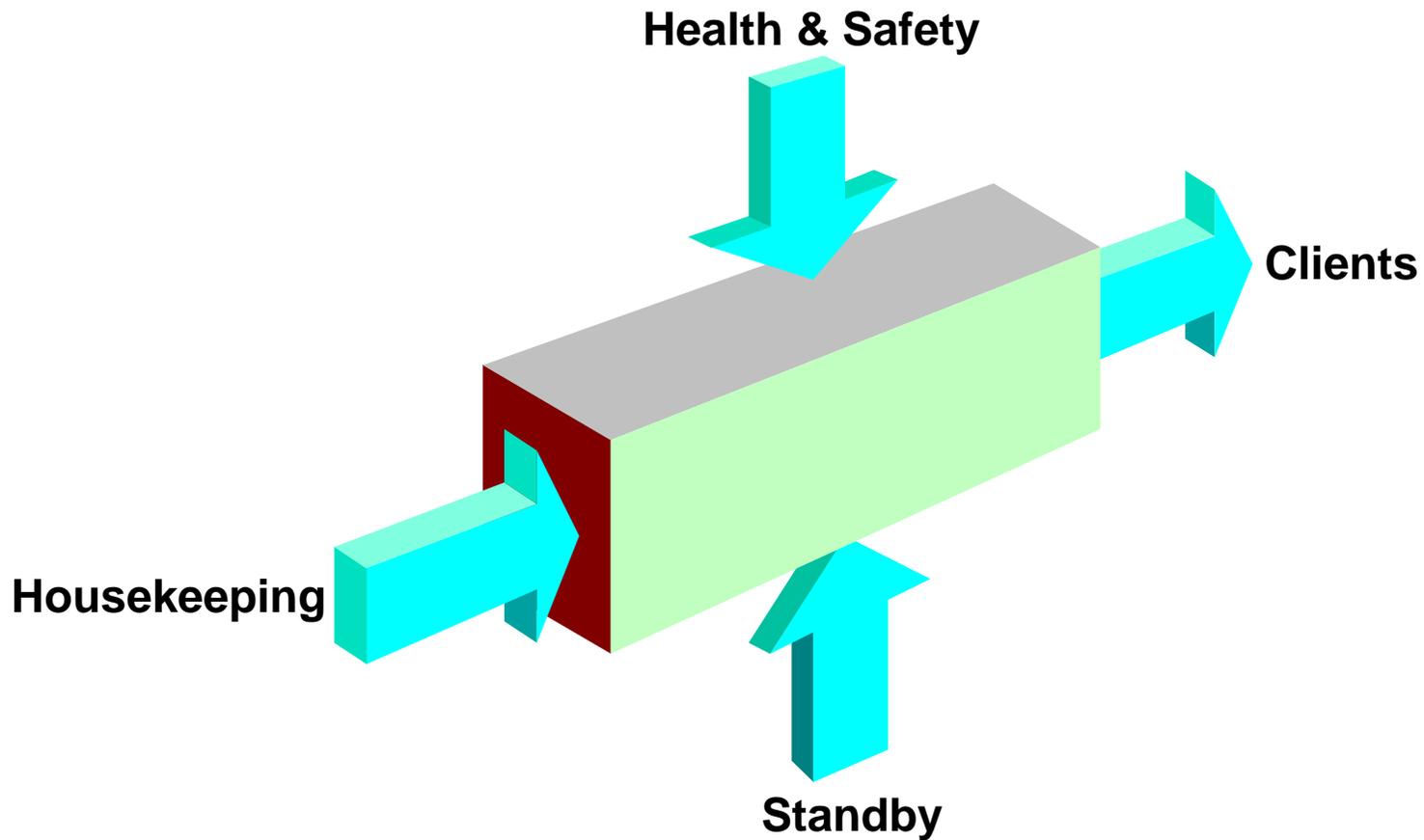
- **Failure of the Housekeeping stream would not affect the Health & Safety or Standby streams**

**Multiple decommutation processes can feed a single parameter server**

- **Housekeeping, Health & Safety and Standby are separate decommutation processes that are producers for a single parameter server**
  - **Housekeeping and Standby produce parameters simultaneously for the same parameter server**
  - **Parameter server consumer clients can use the same parameter server if a switch from Housekeeping to Health & Safety occurs**

# Telemetry Subsystem Stream Oriented Interface

---



# Telemetry Decommutation

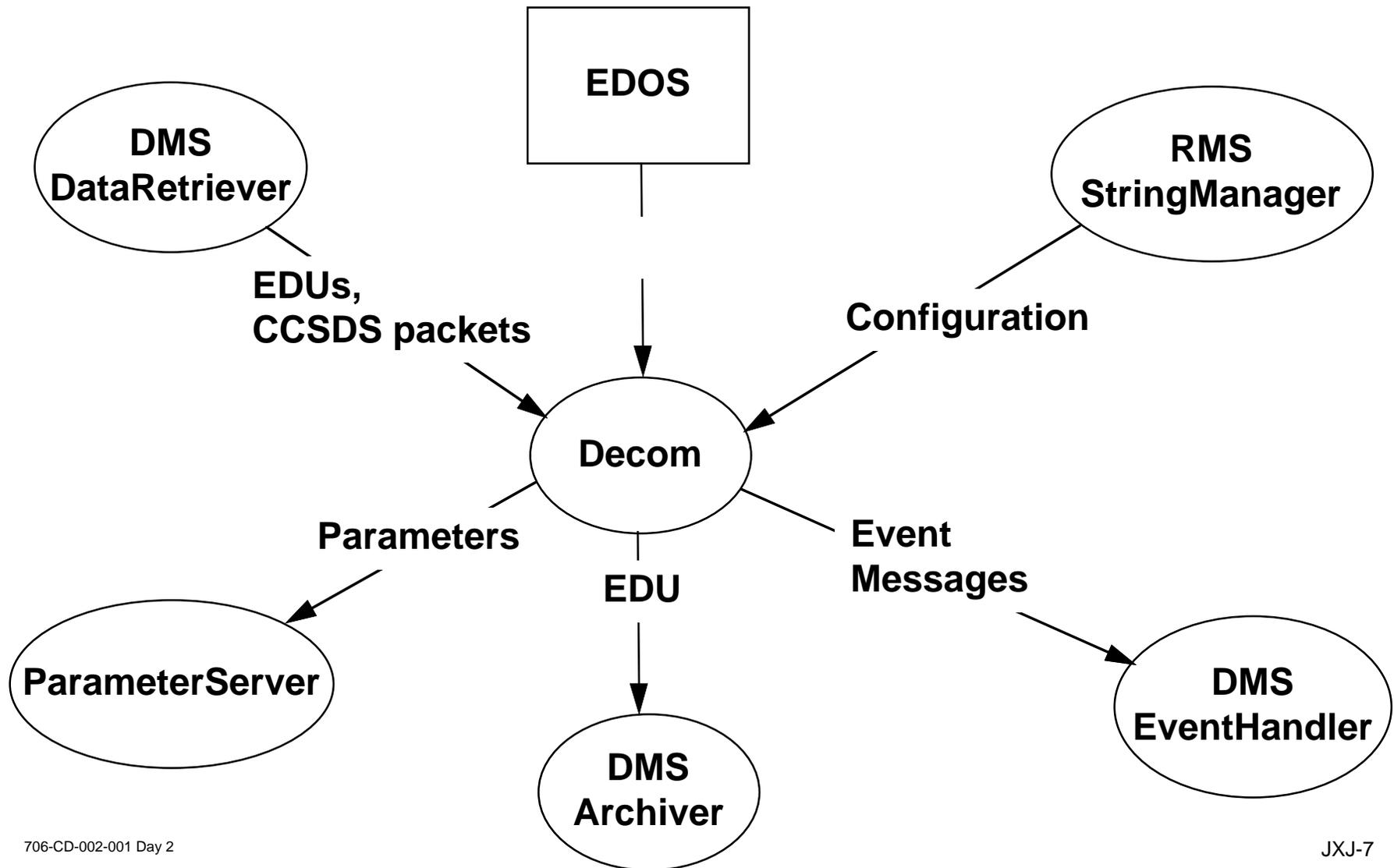
---

The Telemetry Subsystem has 3 components:

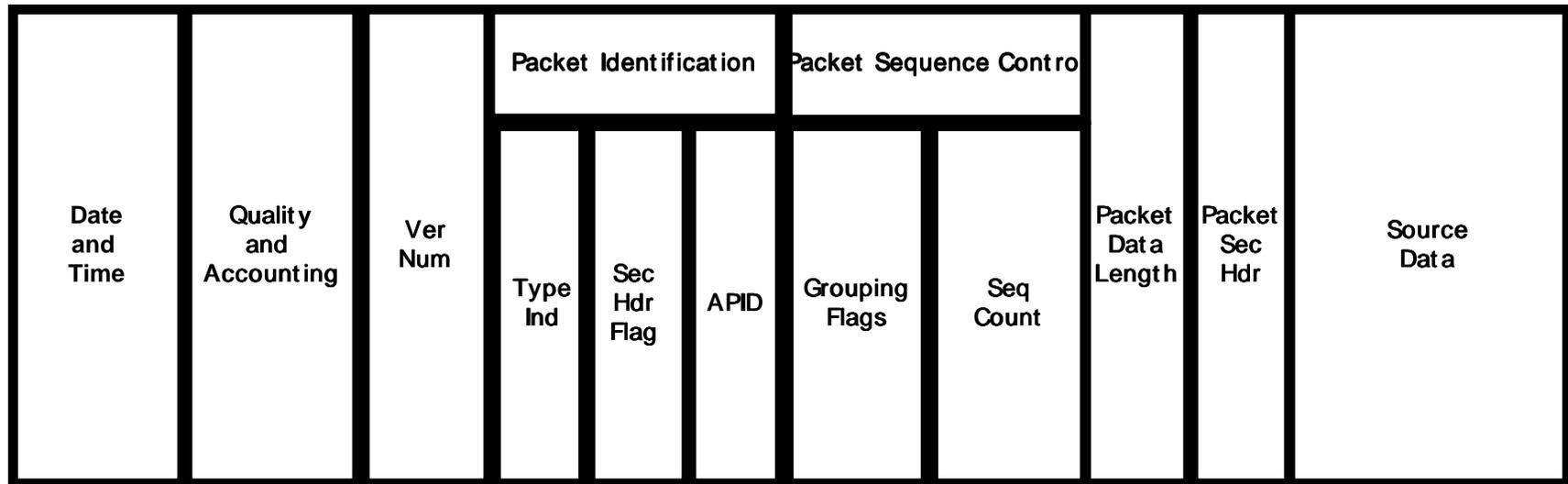
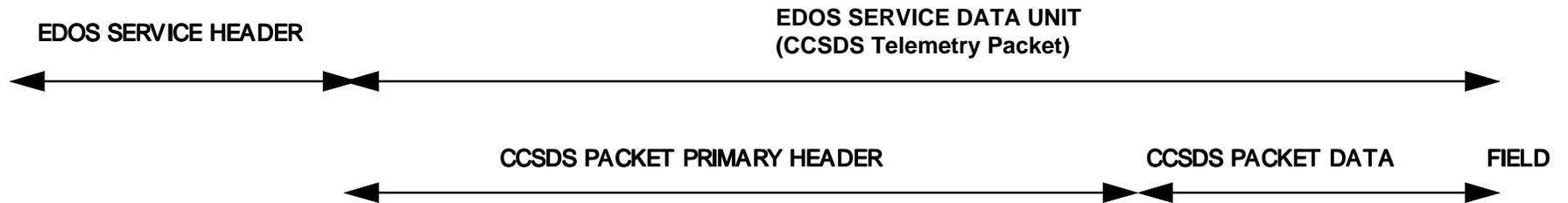
## Telemetry Decommutation

- Memory Dump
- Spacecraft State Check

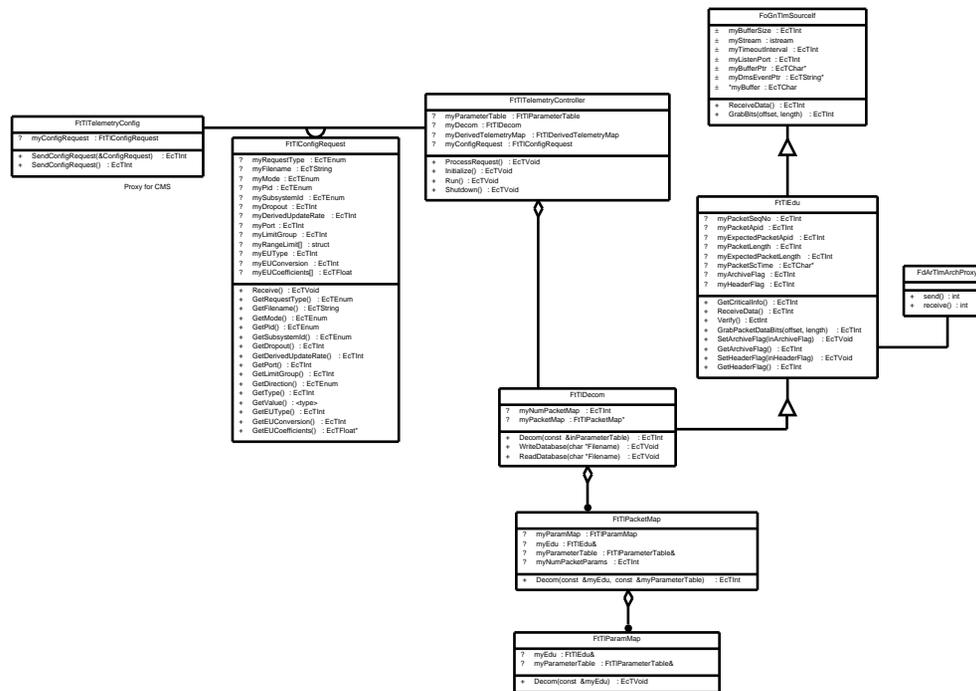
# Telemetry Decommutation Process Context Diagram



# Telemetry Decommutation EDOS Data Unit

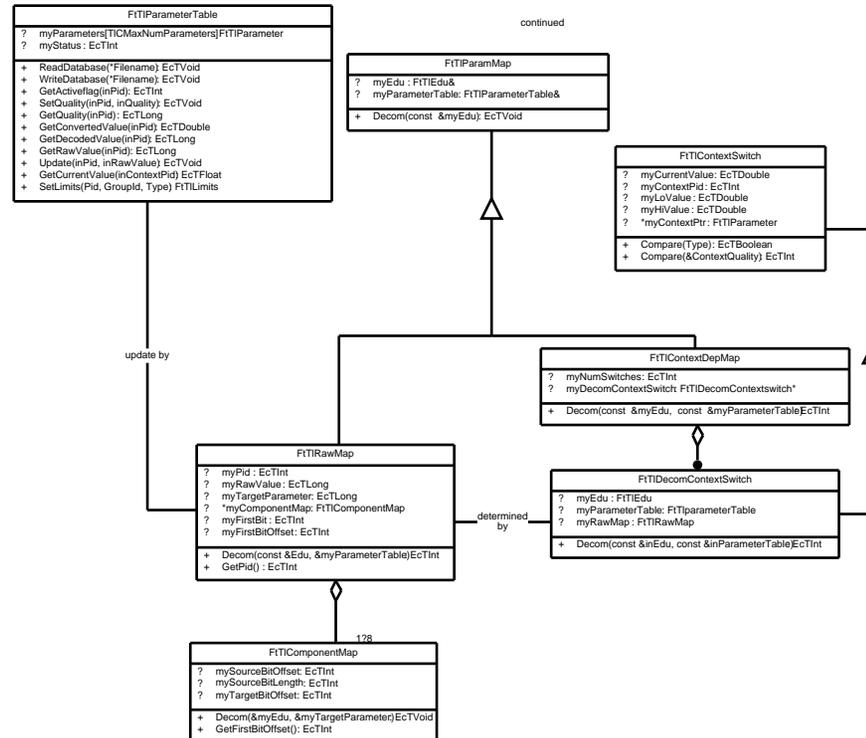


# Telemetry Decommutation Object Model (1 of 2)

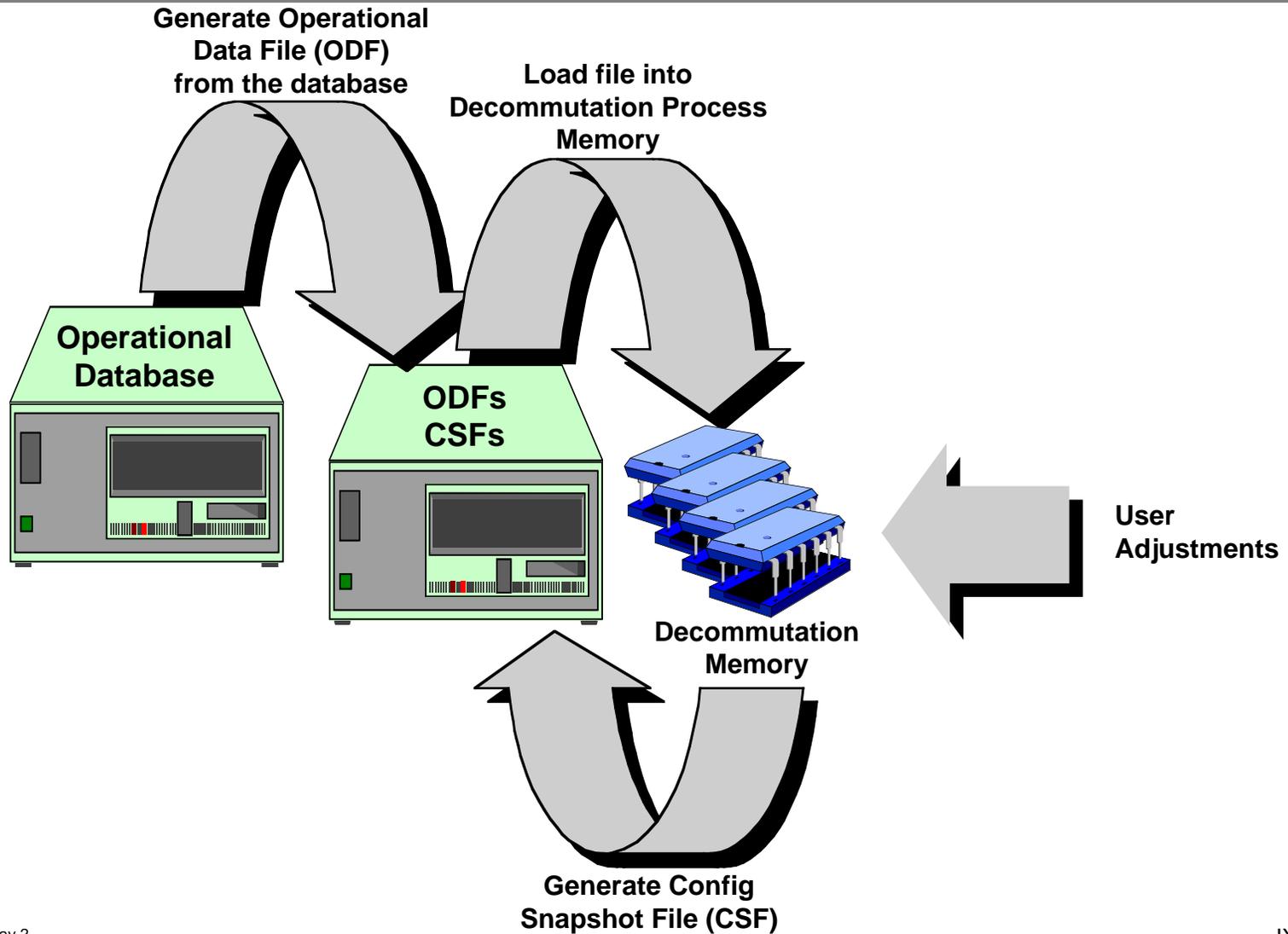


continued

# Telemetry Decommutation Object Model (2 of 2)



# Telemetry Decommuration Operational Data File Generation



# Telemetry Decommulation

## Key Features

---

**Telemetry decommutation is database driven**

- **Operational data file is generated from the database**
  - **Reflects the two main branches of the decommutation model**

**Parameter mappings**

**Parameter processing**

- **The operational data file is generated once to be used every time the decommutation process is started**
- **All parsing of the operational data file is performed at generation time to reduce start-up time**
- **The process can write out a config snapshot file for:**
  - **Mirroring the decommutation process**
  - **Saving the current decommutation settings for later use**

**Scalable design allows for resizing to a larger or smaller scale**

- **Telemetry decommutation is scalable through database modification**
  - **Can add or subtract parameters, limits, coefficients, and derived parameter equations**

# Telemetry Decommutation

## Key Features (continued)

---

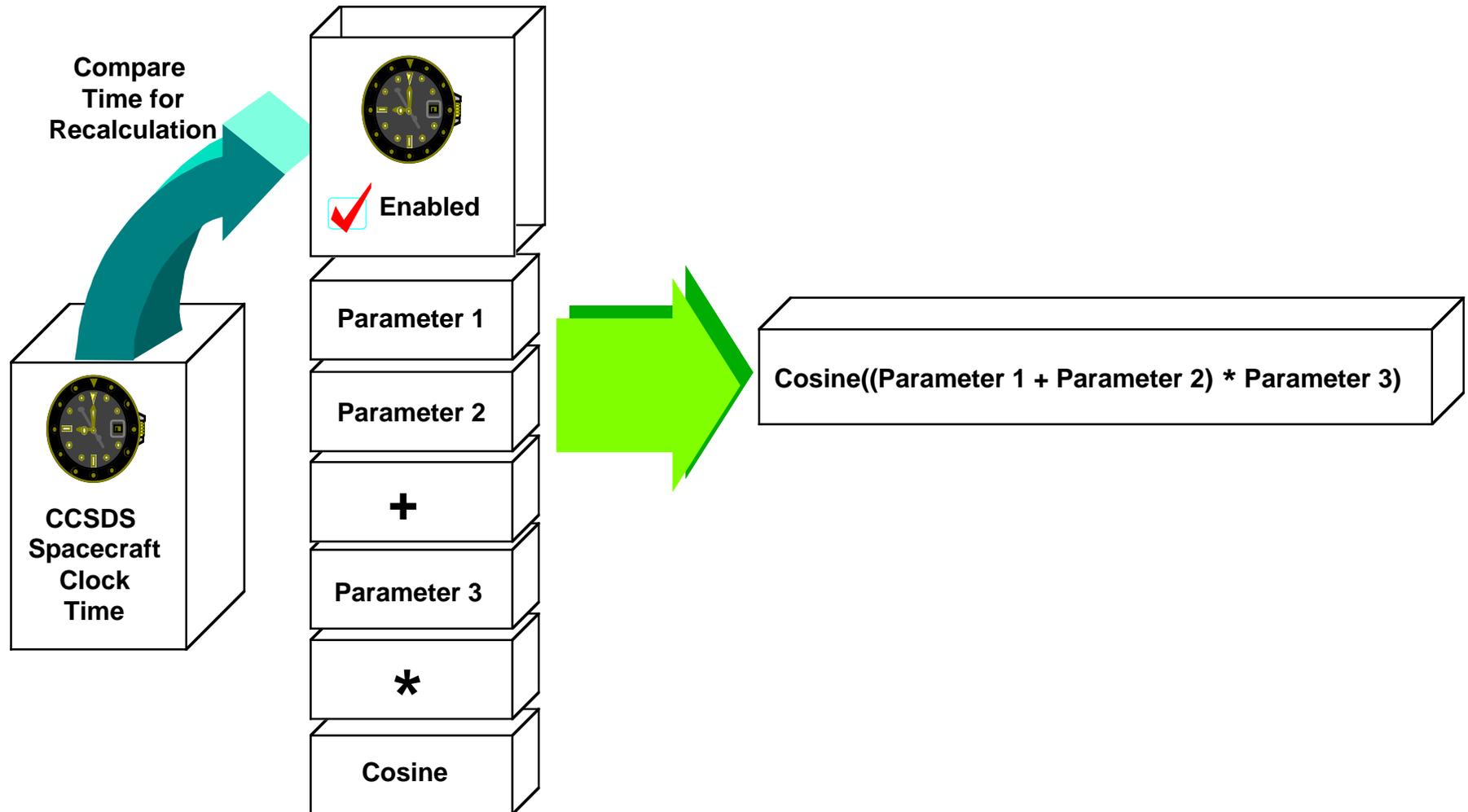
**Capable of dynamically adjusting current values of limits and EU algorithm coefficients loaded from the operational data file**

### **Selective decommutation**

- **Can be selected on all parameters, a predefined group of parameters, or a single parameter**
- **Dynamic selection at the user's request**
- **Adjusting selective decommutation does not require a database change**
- **Maintains a count of selected derived equation dependencies to determine if the parameter is selected for decommutation**

### **Exponential EU Conversion (AM-1 specific)**

# Decommutation: Derived Parameters Recalculation



# Decommutation: Derived Parameters Key Features

---

**Evaluation interval for recalculation is based on the spacecraft clock time extracted from the CCSDS secondary header**

- **Uncouples the recalculation rate from the line transmission rate**

**Selective calculation of derived parameters**

- **Derived parameter processing can be enabled or disabled for all derived parameters**
- **Derived parameter processing can be individually enabled or disabled**
- **When derived parameter processing is enabled**
  - **Other derived parameters used in the equation are enabled**
  - **Decommutated parameters used in the equation are selected for decommutation**

# Decommutation: Derived Parameters

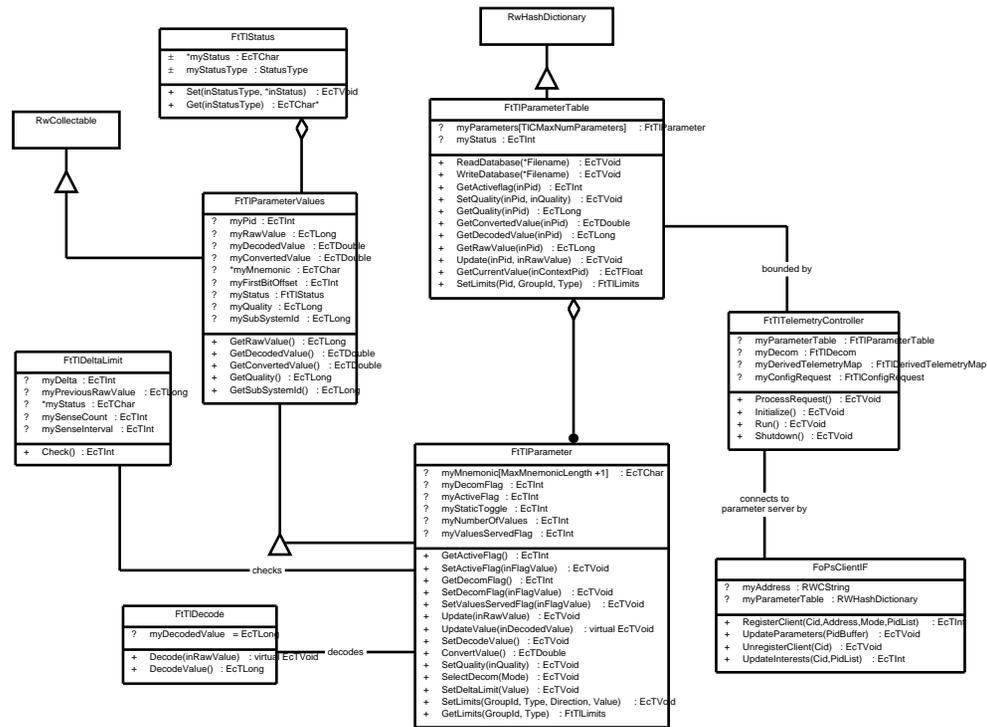
## Key Features (continued)

---

### Equations implemented in Reverse Polish Notation (RPN)

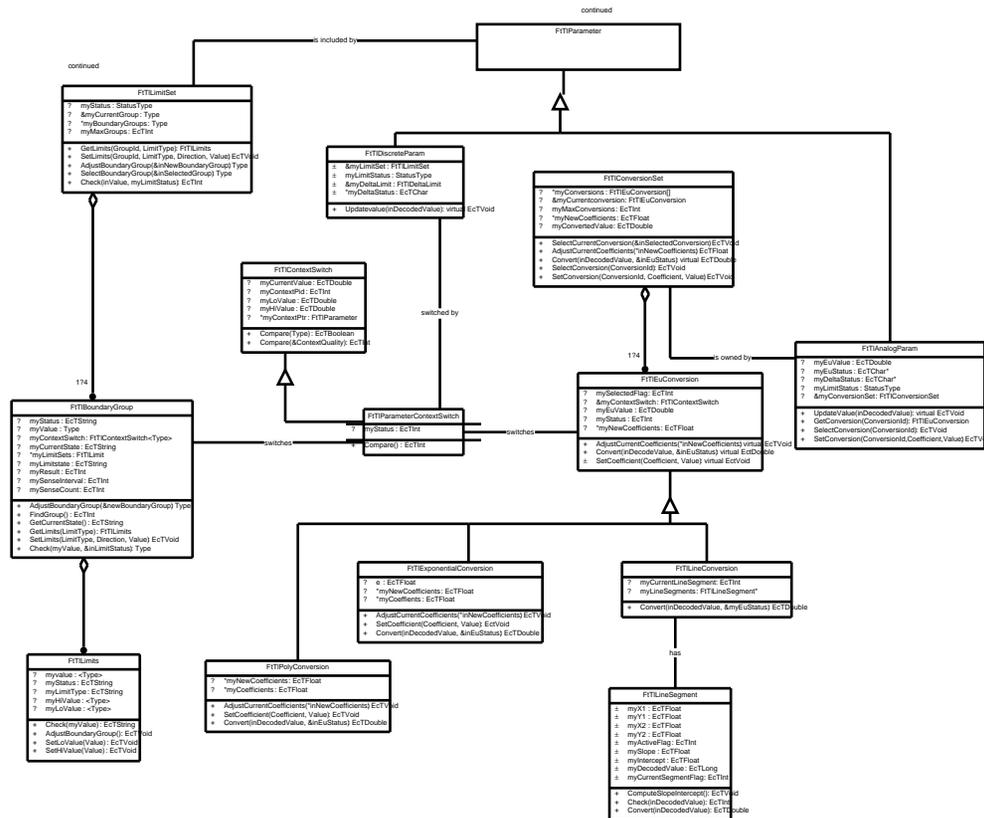
- Precedence implied by order of elements
- The database is in standard notation
  - The operational data file is parsed from the database into RPN
  - Pre-parsing gives a performance increase during run-time
- Virtual functions allow easy expansion of additional operators
- Handles unary and binary operators

# Decommutation Parameter Table Object Model (1 of 2)



continued

# Decommutation Parameter Table Object Model (2 of 2)



# Decommutation: Engineering Unit Conversion Key Features

---

**Allows the user to select and adjust the coefficients of the conversions**

**Maximum number of polynomial coefficients and line segments are database defined**

- **Allows for easy changes of the maximums**

**Scalable design for adding new algorithms through use of virtual functions**

**Converts an analog parameter's value**

- **User selected or context dependent selected conversion algorithm**
- **Multiple EU conversions (up to the database defined maximum) per parameter in any combination of the supported conversion types**

**Analog parameter EU conversion supports:**

- **Polynomial algorithm**                    (  $y = C_0 + C_1x + C_2x^2 + \dots + C_7x^7$  )
  - **Segmented polynomial using context dependent EU conversion**
- **Line segment algorithm**            (  $y = mx + b$  )
- **Exponential algorithm**            (  $y = C_0 + C_1e^{(C_2x)}$  )

**Updates parameter object with EU converted value**

# Decommunation: Limit Checking Key Features

---

- Delta and boundary limit checking on analog and discrete parameters
- Multiple boundary limit groups up to the database defined maximum (which is currently 4) per parameter
  - User selected or context dependent selected
  - Each group contains multiple ranges
    - LoRed, LoYellow, HiYellow, and HiRed limits
    - New ranges (e.g. LoRail and HiRail) can be added via the database
  - Number of groups are adjustable via the database
- Users can adjust the limit values

# Telemetry Decommuration Description

---

## **PART 1: Extraction of the raw value from the data stream**

- **Data ingestion**

- **EDUs from EDOS or DMS (replay)**
- **CCSDS packets from DMS**

**Archive instrument engineering telemetry in CCSDS packet form  
Not required for AM-1**

- **Extracts the raw value for processing in part 3**

## **PART 2: Calculation of derived parameters is based on parameter values**

- **Calculates a value on a timed interval for processing in part 3**

## **PART 3: Processing the parameter value**

- **EU conversion**
- **Range and delta limit checking**
- **Forwards parameter values to the parameter server**

# Memory Dump

---

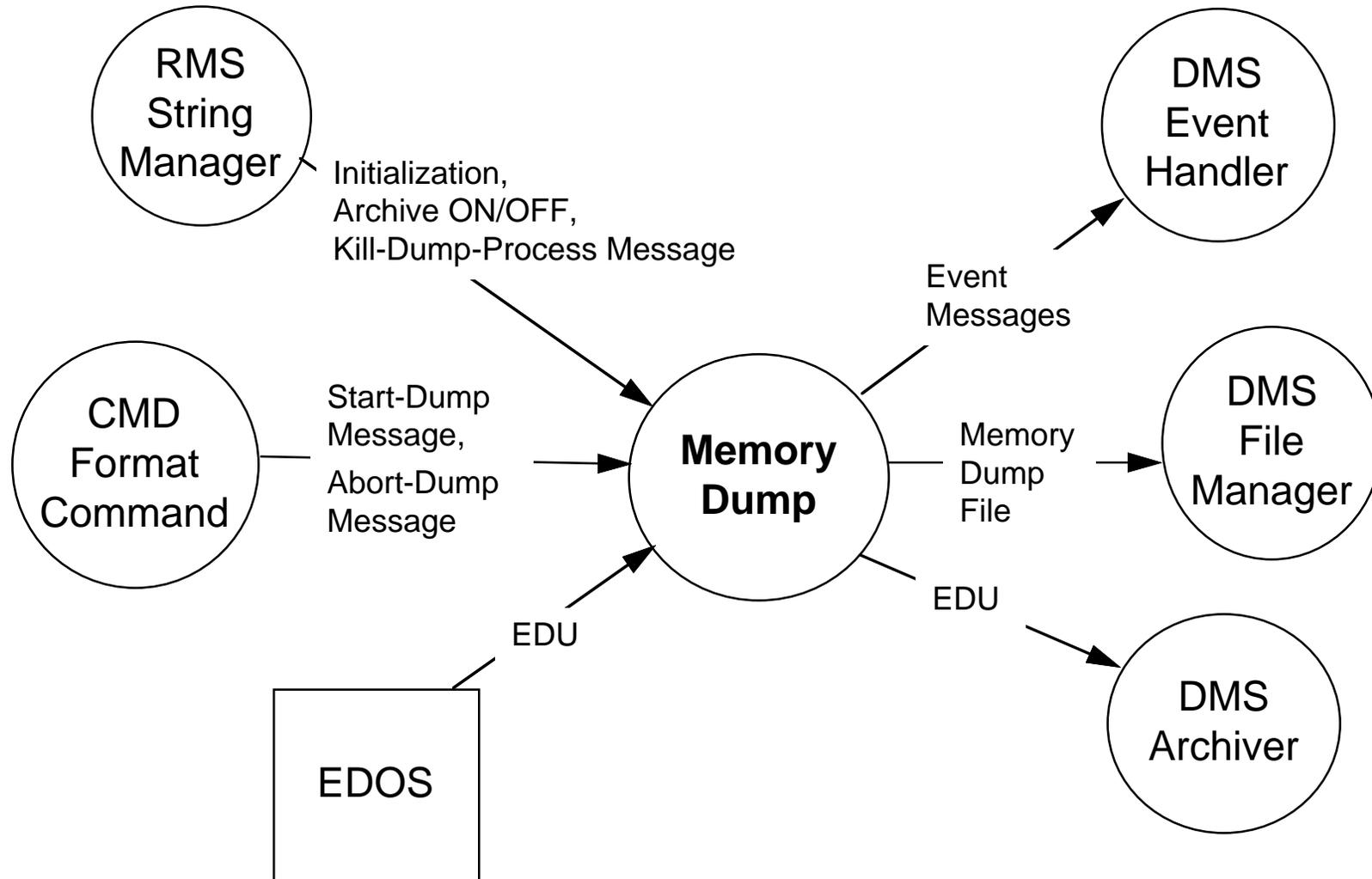
The Telemetry Subsystem has 3 components:

- Telemetry Decommutation

Memory Dump

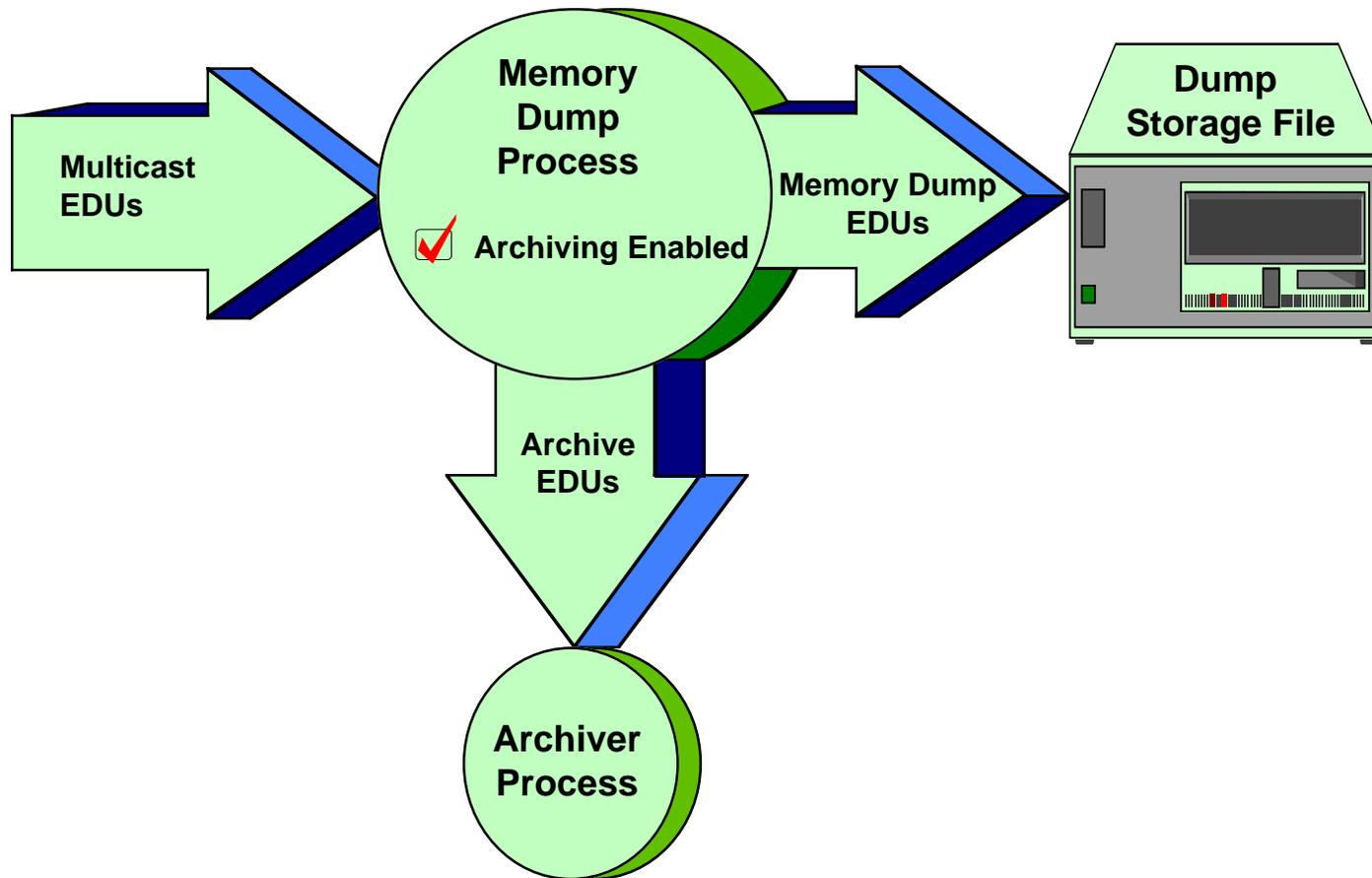
- Spacecraft State Check

# Memory Dump Process Context Diagram



# Memory Dump Data Flow

---



# Memory Dump Key Features

---

The memory dump session can be started anytime before memory dump data is downlinked

- **Memory dump session is started upon the receipt of a start-dump message from the command subsystem**
- **After a memory dump session is started, the process recognizes and generates an event message when the actual memory dump data starts and stops**
  - **Start of memory dump is recognized when the embedded RT address is not 31 (AM-1 specific)**
  - **End of memory dump is recognized when the dump count is satisfied**
- **A memory dump session can be aborted**

# Memory Dump Key Features (continued)

---

**Collects and stores the contents of downlinked spacecraft or instrument computer memory dumps in a local memory dump storage file**

- **The file contains memory dump EDUs (those whose embedded RT address is not 31, AM-1 specific)**
- **The file is forwarded to DMS when a memory dump is complete**
  - **CMS can retrieve the file immediately for processing**

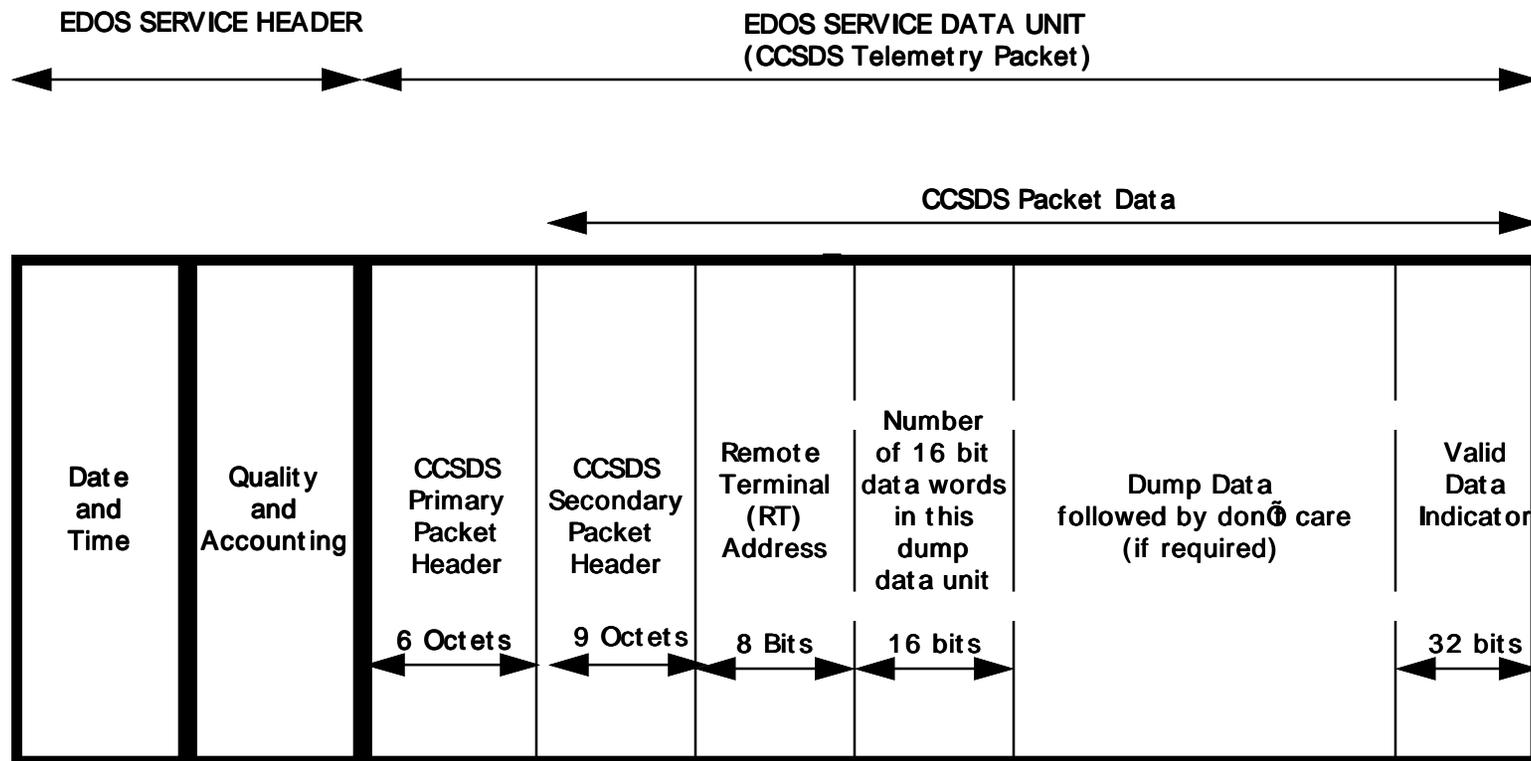
**Easily expanded to allow memory dump processing of replayed telemetry**

**Shares the EDU handler class with telemetry decommutation**

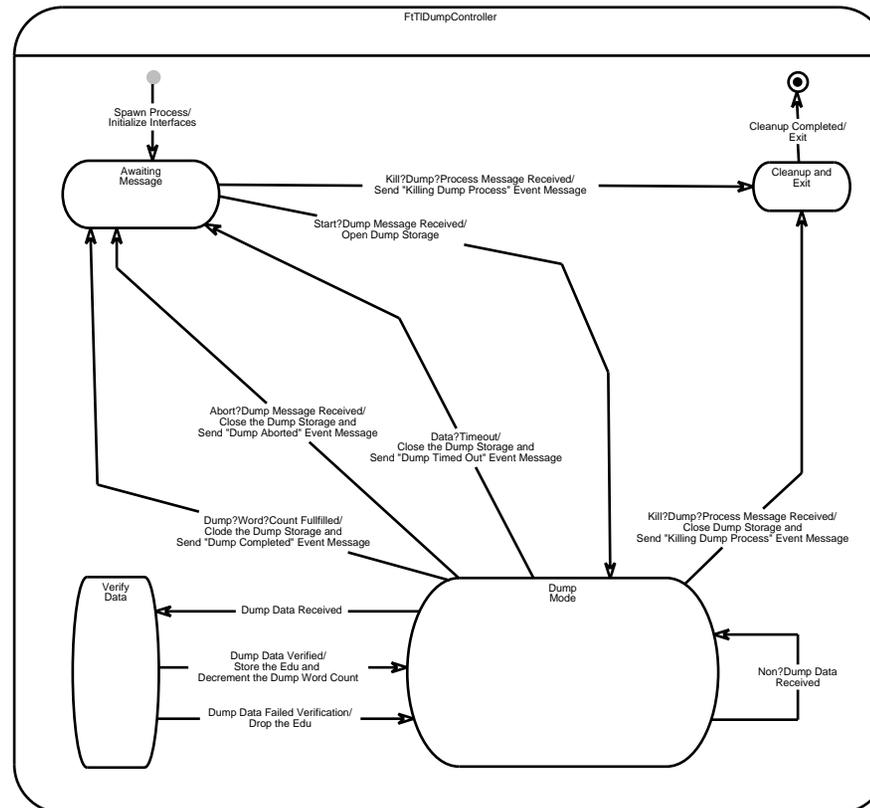


# Memory Dump EDOS Data Unit

## AM-1 EDOS Data Unit Format



# Memory Dump State Transition Diagram



# Memory Dump State Transition Diagram Description

---

- The Memory Dump process goes into the *Awaiting Message State* after process initialization
- *Awaiting Message State:*
  - When a Start-Dump message is received:
    - A dump storage file is opened
    - The Start-Dump message is written to the dump storage file
    - Dump word count is initialized from information in the Start-Dump message
    - The process moves into the *Dump Mode State*
- *Dump Mode State:*
  - When dump data is received (i.e., the embedded RT Address is not 31) the process moves into the *Verify Data State*.
  - When the dump word count goes to zero or a data-timeout occurs or an Abort-Dump message is received:
    - The dump storage file is closed and sent to DMS
    - The process moves into the *Awaiting Message State*

# Memory Dump State Transition Diagram Description (continued)

---

- ***Verify Data State:***
  - Performs a sequence check and generates a sequence error event message if out of sequence
  - If the data's APID does not match the expected APID, or if the data's packet length does not match the expected packet length:
    - The EDU fails to verify and it is dropped
    - The process moves back to the *Dump Mode State*
  - If the dump data verifies:
    - The dump data is written out to the dump storage file
    - The dump word count is decremented
    - The process moves back to the *Dump Mode State*
- Anytime a Kill-Dump-Process message is received the process goes into the *Cleanup and Exit State* to shutdown the process

# Spacecraft State Check

---

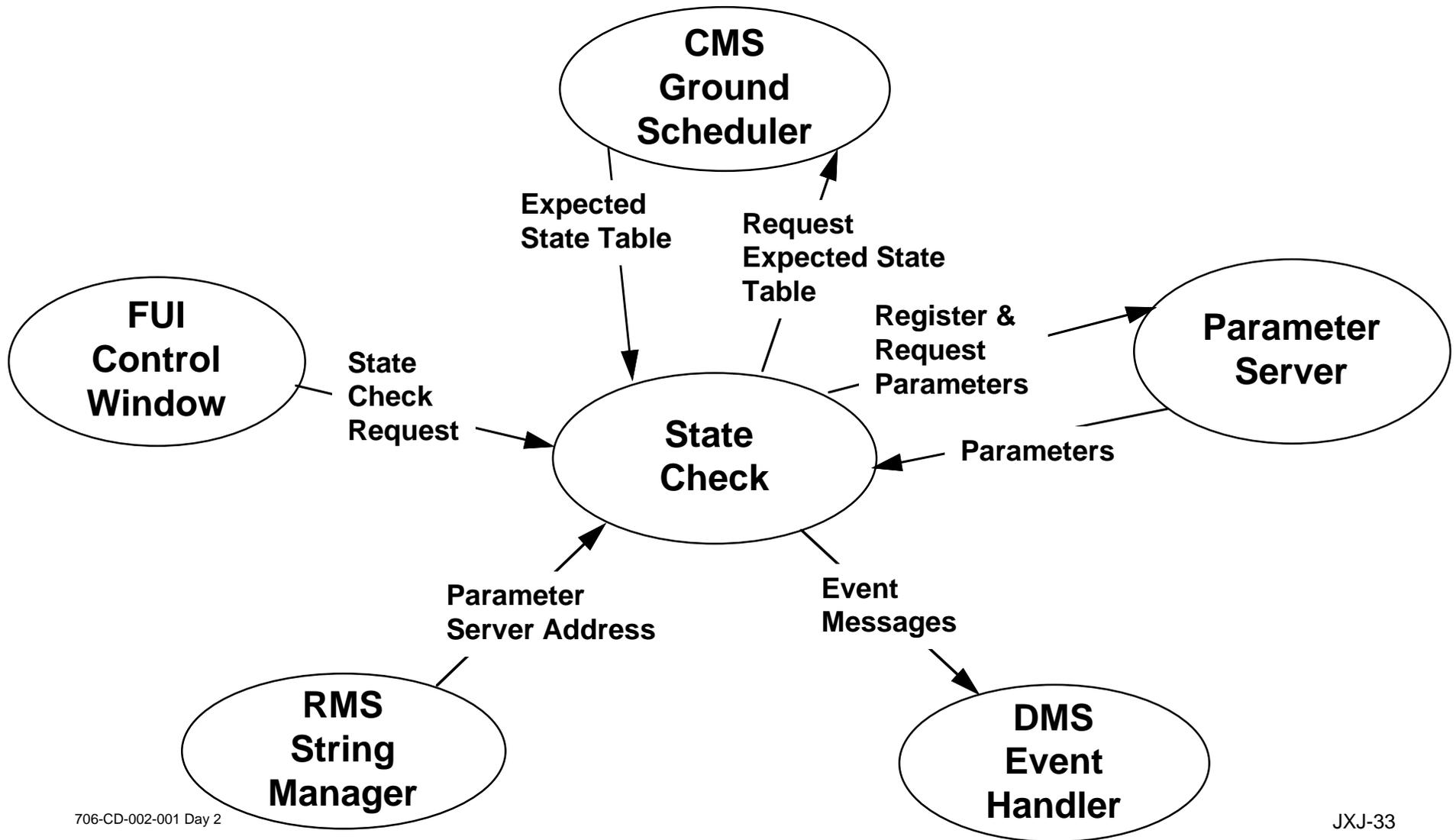
**The Telemetry Subsystem has 3 components:**

- **Telemetry Decommutation**
- **Memory Dump**



# Spacecraft State Check Process Context Diagram

---



# Spacecraft State Check

## Key Features

---

**Verify that the condition of the spacecraft at start of contact is the same as the predicted condition using expected back orbit command execution**

**Expected state table is dynamically generated by CMS on request from ground scripts or by the user**

- **Typically generated for each scheduled spacecraft contact**
- **Contains:**
  - **List of telemetry parameter identifiers**
  - **Expected decoded value of each parameter**

**Permits state check request prior to spacecraft contact**

- **Process waits for the decommutation of a full master cycle before state check or baseline is executed**

**Provides a mechanism to recheck condition anytime during contact**

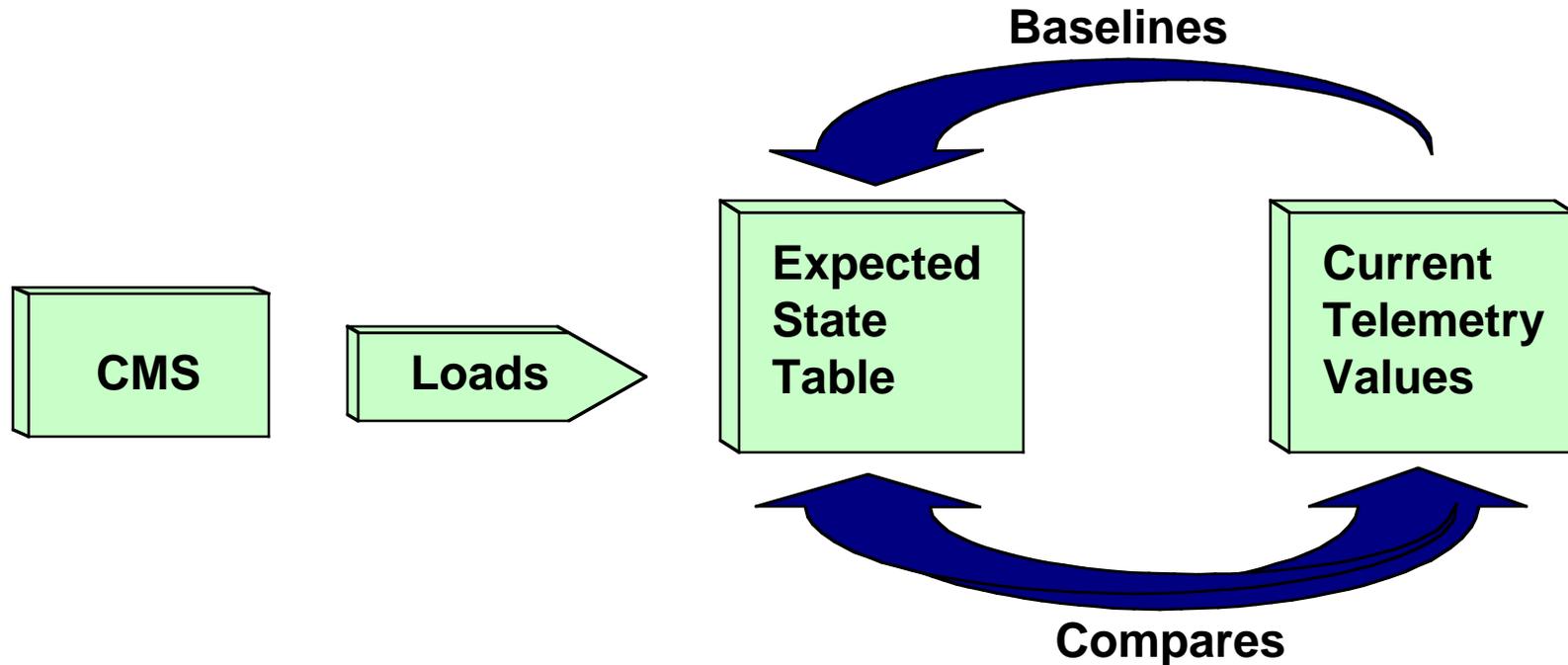
- **Typically after a baseline has been performed**

**Baselines current telemetry values into expected state table**

- **Values in the expected state table replaced with current values**
- **State check can be performed on this new expected state table**

# Spacecraft State Check Expected State Table

---



# Spacecraft State Check User Directives

---

## **Load state check service**

- **Passes load request from FUI to CMS**
- **CMS dynamically generates expected state table and supplies it to spacecraft state check**

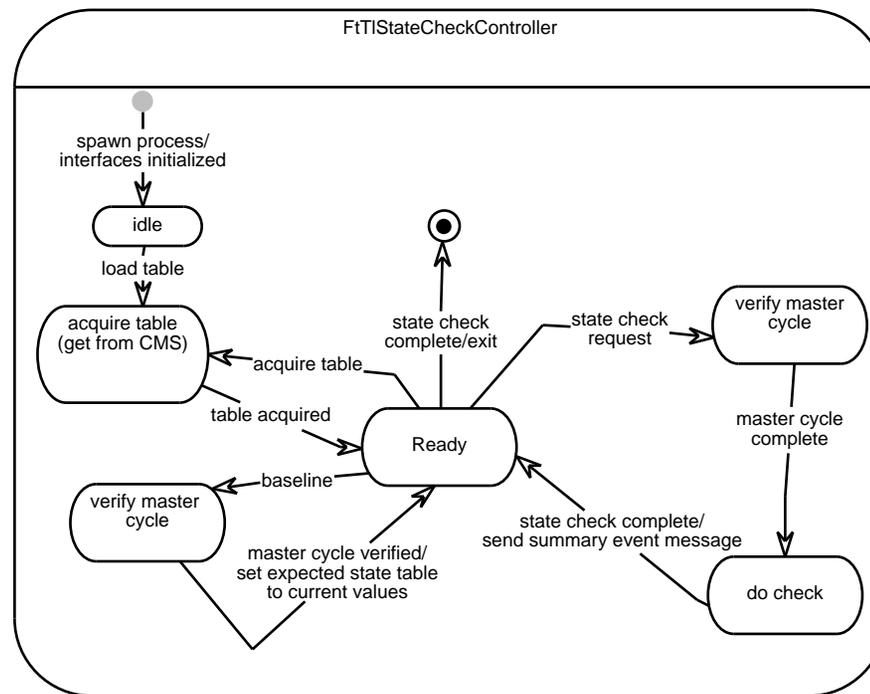
## **Baseline state check service**

- **Replaces values in expected state table with current telemetry values**

## **Perform state check service**

- **Compare current telemetry values retrieved from the parameter server against the expected state table**
- **Generate event message for each miscompare**
- **Generate summary event message at end of state check**

# Spacecraft State Check State Transition Diagram



# Spacecraft State Check State Transition Diagram Description

---

The State Check process goes into the *Idle State* when initialization is complete

## *Idle State*

- Acquires expected state table from CMS when Load Table request is received and moves to *Ready State*

## Ready State

- Determines request type (state check or baseline)
- If request is baseline, moves to *Verify Master Cycle State*
  - When verification is done, replaces expected state table with current telemetry values
- If request is a state check, moves to *Verify Master Cycle State*
  - When verification is done moves into *Do Check State*
- When a Kill-Process is received, the state check process exits

# Spacecraft State Check State Transition Diagram Description (continued)

---

- ***Verify Master Cycle State***
  - Verifies the decommutation of a complete master cycle before initiating state check or baseline request
  - If baseline request:
    - Replaces values in expected state table with current telemetry values
    - Process returns to *Ready State*
  - If State Check request:
    - Moves to *Do Check State*
- ***Do Check State***
  - Compare values in expected state table with current telemetry values
  - Generates event message for each miscompare
  - Generates summary event message at end of state check
  - Returns to *Ready State*